

Python /  
Numpy

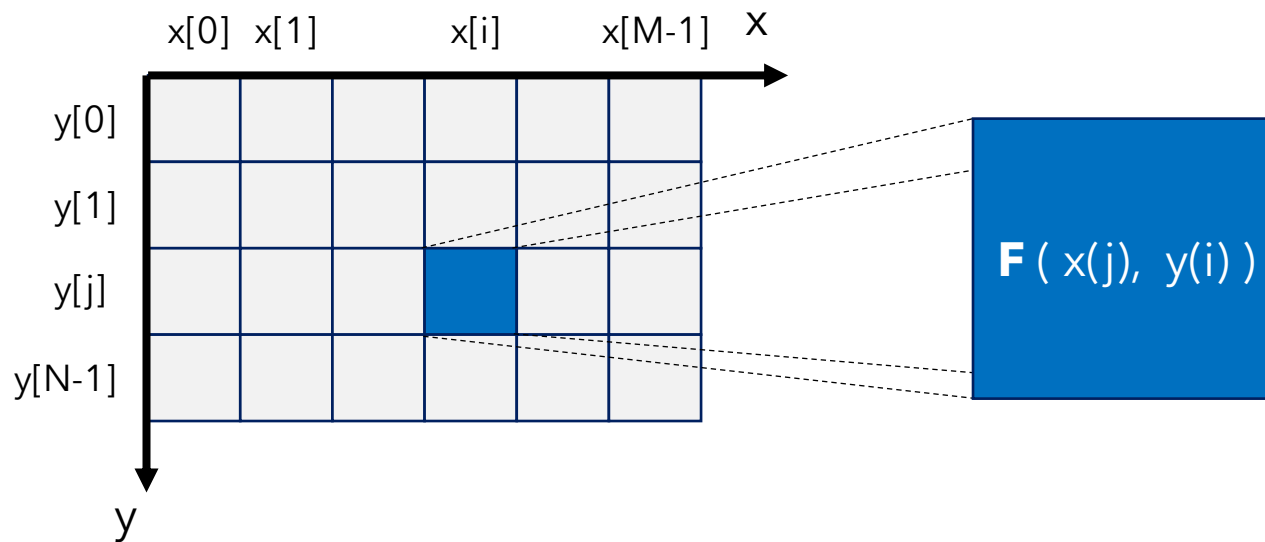
Meshgrid

---

Digital Methods  
Institut d'Optique / Notions

# How to fill a 2D array ?

- Problem :
  - Fill a **2D array** with a specific value depending on x-axis and y-axis index



- For example :

```
def F(a, b):  
    return a + b
```

- We assume that x and y are defined :

```
x = np.arange(...) # length = M  
y = np.arange(...) # length = N
```

# How to fill a 2D array ?

- First idea : a **double loop** on  $i$  and  $j$

```
output_array = np.zeros((N, M))

for i in range(N):
    for j in range(M):
        output_array[i][j] = F(x[j], y[i])
```

# How to fill a 2D array ?

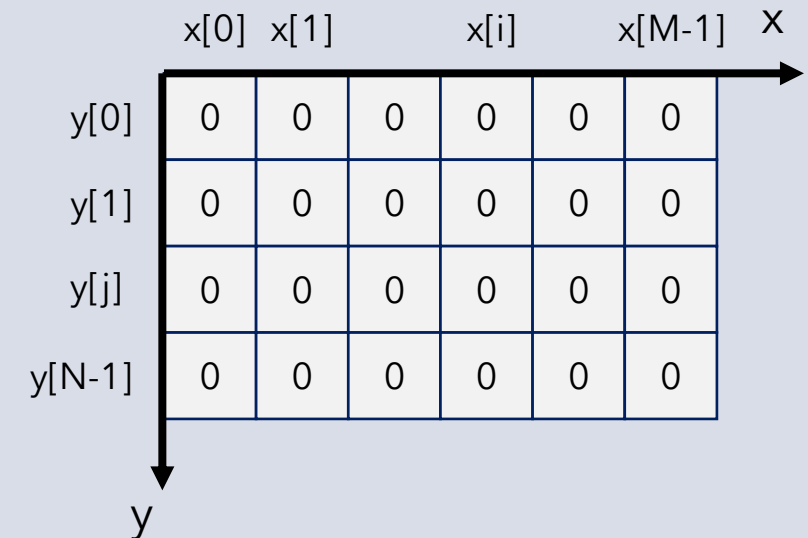
- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```



# How to fill a 2D array ?

- Example (N and M are integers) :

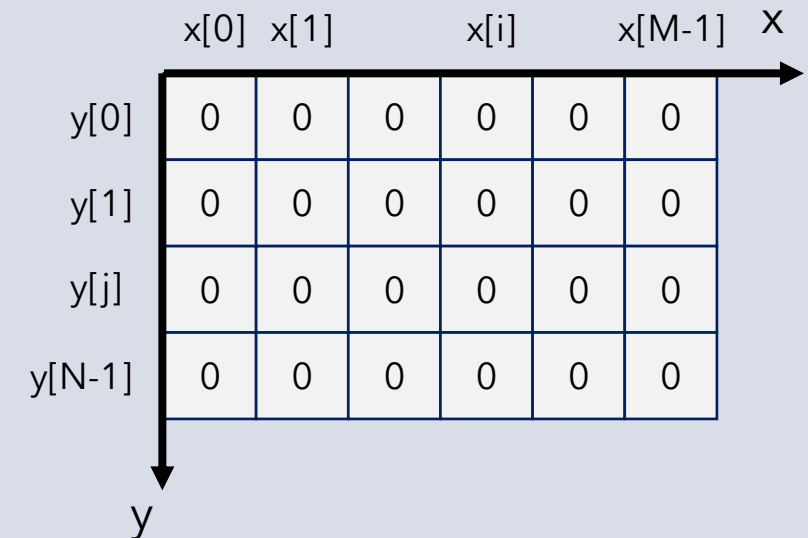
```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

**i = 0**

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```



# How to fill a 2D array ?

- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

$i = 0$

$j = 0 \rightarrow \text{output\_array}[0][0] = F(x[0], y[0]) = 0$

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

	x[0]	x[1]	x[i]	x[M-1]	x
y[0]	0	0	0	0	0
y[1]	0	0	0	0	0
y[j]	0	0	0	0	0
y[N-1]	0	0	0	0	0

The diagram shows a 2D grid representing the output array. The horizontal axis is labeled 'x' and the vertical axis is labeled 'y'. The columns are labeled x[0], x[1], x[i], and x[M-1]. The rows are labeled y[0], y[1], y[j], and y[N-1]. The cell at the intersection of x[0] and y[0] is highlighted in yellow and contains the value 0. All other cells in the grid contain the value 0.

# How to fill a 2D array ?

- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

$i = 0$

$j = 0 \rightarrow \text{output\_array}[0][0] = F(x[0], y[0]) = 0$

$j = 1 \rightarrow \text{output\_array}[0][1] = F(x[1], y[0]) = 1$

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

	x[0]	x[1]	x[i]	x[M-1]	x	
y[0]	0	1	0	0	0	0
y[1]	0	0	0	0	0	0
y[j]	0	0	0	0	0	0
y[N-1]	0	0	0	0	0	0

The diagram shows a 2D array with rows labeled y[0], y[1], y[j], and y[N-1] and columns labeled x[0], x[1], x[i], and x[M-1]. The x-axis is labeled 'x' and the y-axis is labeled 'y'. The cell at row y[0] and column x[1] is highlighted in yellow and contains the value 1. All other cells in the array contain the value 0.

# How to fill a 2D array ?

- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

$i = 0$

$j = 0 \rightarrow \text{output\_array}[0][0] = F(x[0], y[0]) = 0$

$j = 1 \rightarrow \text{output\_array}[0][1] = F(x[1], y[0]) = 1$

$j = 2 \rightarrow \text{output\_array}[0][2] = F(x[2], y[0]) = 2$

$j = 3 \rightarrow \text{output\_array}[0][3] = F(x[3], y[0]) = 3$

$j = 4 \rightarrow \text{output\_array}[0][4] = F(x[4], y[0]) = 4$

$j = 5 \rightarrow \text{output\_array}[0][5] = F(x[5], y[0]) = 5$

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

	x[0]	x[1]	x[i]	x[M-1]	x	
y[0]	0	1	2	3	4	5
y[1]	0	0	0	0	0	0
y[j]	0	0	0	0	0	0
y[N-1]	0	0	0	0	0	0



# How to fill a 2D array ?

- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

$i = 0$

...

$i = 1$

$j = 0 \rightarrow \text{output\_array}[1][0] = F(x[0], y[1]) = 1$

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

	x[0]	x[1]	x[i]	x[M-1]	x	
y[0]	0	1	2	3	4	5
y[1]	1	0	0	0	0	0
y[j]	0	0	0	0	0	0
y[N-1]	0	0	0	0	0	0

The table illustrates the output of the function F(a, b) = a + b applied to a 2D grid of coordinates (x, y). The x-axis is labeled 'x' and the y-axis is labeled 'y'. The grid has columns for x[0], x[1], x[i], and x[M-1], and rows for y[0], y[1], y[j], and y[N-1]. The value 1 is highlighted in the cell at (x[0], y[1]), representing the result of F(x[0], y[1]) = 0 + 1 = 1. All other cells contain 0, representing the result of F(x[j], y[i]) = x[j] + y[i] = 0 + 0 = 0.

# How to fill a 2D array ?

- Example (N and M are integers) :

```
output_array = np.zeros((N, M))
```

```
for i in range(N):  
    for j in range(M):  
        output_array[i][j] = F(x[j], y[i])
```

**i = 0**

...

**i = 1**

$j = 0 \rightarrow \text{output\_array}[\mathbf{1}][0] = F(x[0], y[\mathbf{1}]) = 1$

$j = 1 \rightarrow \text{output\_array}[\mathbf{1}][1] = F(x[1], y[\mathbf{1}]) = 2$

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

	x[0]	x[1]	x[i]	x[M-1]	x	
y[0]	0	1	2	3	4	5
y[1]	1	2	0	0	0	0
y[j]	0	0	0	0	0	0
y[N-1]	0	0	0	0	0	0

The table illustrates the output of the function F(a, b) = a + b for a 2D array. The x-axis is labeled 'x' and the y-axis is labeled 'y'. The values in the table are the result of F(x[j], y[i]). The cell at (y[1], x[1]) is highlighted in yellow, showing the value 2. The cell at (y[1], x[0]) is highlighted in grey, showing the value 1. The cell at (y[0], x[0]) is highlighted in grey, showing the value 0. The cell at (y[0], x[1]) is highlighted in grey, showing the value 1. The cell at (y[0], x[2]) is highlighted in grey, showing the value 2. The cell at (y[0], x[3]) is highlighted in grey, showing the value 3. The cell at (y[0], x[4]) is highlighted in grey, showing the value 4. The cell at (y[0], x[5]) is highlighted in grey, showing the value 5. The cell at (y[1], x[0]) is highlighted in grey, showing the value 1. The cell at (y[1], x[1]) is highlighted in yellow, showing the value 2. The cell at (y[1], x[2]) is highlighted in grey, showing the value 0. The cell at (y[1], x[3]) is highlighted in grey, showing the value 0. The cell at (y[1], x[4]) is highlighted in grey, showing the value 0. The cell at (y[1], x[5]) is highlighted in grey, showing the value 0. The cell at (y[j], x[0]) is highlighted in grey, showing the value 0. The cell at (y[j], x[1]) is highlighted in grey, showing the value 0. The cell at (y[j], x[2]) is highlighted in grey, showing the value 0. The cell at (y[j], x[3]) is highlighted in grey, showing the value 0. The cell at (y[j], x[4]) is highlighted in grey, showing the value 0. The cell at (y[j], x[5]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[0]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[1]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[2]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[3]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[4]) is highlighted in grey, showing the value 0. The cell at (y[N-1], x[5]) is highlighted in grey, showing the value 0.

# How to fill a 2D array ?

- Second method : using **Numpy arrays** methods (*meshgrid*)

```
XX, YY = np.meshgrid(x, y)
```

```
output_array = F(YY, XX)
```

# How to fill a 2D array ?

- Example (N and M are integers) :

```
XX, YY = np.meshgrid(x, y)
```

```
output_array = F(YY, XX)
```

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

XX

x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]

YY

y[0]	y[0]		y[0]		y[0]
y[1]	y[1]		y[1]		y[1]
y[i]	y[i]		y[i]		y[i]
y[N-1]	y[N-1]		y[N-1]		y[N-1]

# How to fill a 2D array ?

- Example (N and M are integers) :

```
XX, YY = np.meshgrid(x, y)
```

```
output_array = F(YY, XX)
```

```
x = np.linspace(0, M-1, M)  
y = np.linspace(0, N-1, N)
```

```
def F(a, b):  
    return a + b
```

XX

x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]
x[0]	x[1]		x[j]		x[M-1]

YY

y[0]	y[0]		y[0]		y[0]
y[1]	y[1]		y[1]		y[1]
y[i]	y[i]		y[i]		y[i]
y[N-1]	y[N-1]		y[N-1]		y[N-1]

	y[0]	y[1]		y[i]		y[N-1]	y
x[0]	0	1	2	3	4	5	
x[1]	1	2	3	4	5	6	
x[j]	2	3	4	5	6	7	
x[M-1]	3	4	5	6	7	8	

x

# How to fill a 2D array ?

- Comparison / Execution time\*

	M=3 / N=5	M=30 / N=50	M=300 / N=500	M=3k / N=5k
<pre>output_array = np.zeros((N, M))  for i in range(N):     for j in range(M):         output_array[i][j] = F(x[j], y[i])</pre>	~ 9 us	690 us	70 ms	7 s
<i>Memory Use</i>	1 x M x N			
<pre>XX, YY = np.meshgrid(x, y)  output_array = F(YY, XX)</pre>	~ 19 us	~ 21 us	~ 1 ms	9.4 ms
	3 x M x N			

\* Executed on the same computer / Core i7 / 16 Go RAM